

DTIC FILE COPY

**Navy Personnel Research and Development Center**

San Diego, CA 92152-0000 TR 89-2 December 1988



2

AD-A202 153

**Steamer II:  
Steamer Prototype Component Inventory  
and User Interface Commands**

Approved for public release; distribution is unlimited.

DTIC  
ELECTE  
S JAN 9 1989 D  
a  
H

89 1 09 172



**Steamer II:  
Steamer Prototype Component Inventory and  
User Interface Commands**

Janet L. Dickieson  
Walter F. Thode  
Navy Personnel Research and Development Center

Kent Newbury  
Systems Engineering Associates  
San Diego, California 92109

Reviewed and approved by  
J. C. McLachlan  
Director, Training Systems Department

Released by  
B. E. Bacon  
Captain, U.S. Navy  
Commanding Officer  
and  
J. S. McMichael  
Technical Director

Approved for public release;  
distribution is unlimited.

Navy Personnel Research and Development Center  
San Diego, California 92152-6800



## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)  NPRDC TR 89-2			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Navy Personnel Research and Development Center		6b. OFFICE SYMBOL (if applicable) Code 14		7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code)  San Diego, California 92152-6800				7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Chief of Naval Operations		8b. OFFICE SYMBOL (if applicable) OP-01		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)  Washington, DC 20350-2000				10. SOURCE OF FUNDING NUMBERS	
				PROGRAM ELEMENT NO 0603720N	PROJECT NO Z1772
11. TITLE (Include Security Classification)  Steamer II: Steamer Prototype Component Inventory and User Interface Commands					
12. PERSONAL AUTHOR(S) Dickieson, J. L., Thode, W. F. (NPRDC), and Newbury, K. (Systems Engineering Associates)					
13a. TYPE OF REPORT Technical Report		13b. TIME COVERED FROM FY88 TO FY89		14. DATE OF REPORT (Year, Month, Day) 1988 December	
15. PAGE COUNT 57					
16. SUPPLEMENTARY NOTATION					
17. COSAT CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)  Simulation, steam plant propulsion, artificial intelligence, Steamer		
FIELD	GROUP	SUB-GROUP			
05	09				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)  Over the last several years, Navy Personnel Research and Development Center produced a Steamer prototype simulation of a 1200-psi steam plant. This simulation is installed on an expensive Symbolics minicomputer at the Surface Warfare Officers School Pacific (SWOSCOLPAC), Coronado, CA. Documentation of the Steamer prototype components and the user interface was needed.  Careful examination of the actual program code provided an inventory that describes the hardware, system software, and application software and list the documentation for the prototype Steamer system. Similarly, systematic exercising of all menu options produced an inventory of the user interface commands.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED / UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS				21. ABSTRACT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>	
22a. NAME OF RESPONSIBLE INDIVIDUAL Janet L. Dickieson				22b. TELEPHONE (Include Area Code) (619) 553-7712	
				22c. OFFICE SYMBOL Code 14	



## FOREWORD

This research was funded as part of the Advanced Development project entitled Advanced Computer-aided Instruction (Program Element Number 0630720N, Work Unit Number Z1772-WX-35042). It was sponsored by the Chief of Naval Operations (OP-01). Systems Engineering Associates performed part of this work under contract N00123-85-D-0151.

The fundamental research goal of the Steamer prototype system was to evaluate the potential of new artificial intelligence (AI) hardware and software technology for supporting the construction of computer-based training systems using graphic representations of complex, dynamic systems. The initial focus is on the propulsion engineering domain. Technologies successfully applied to this domain are expected to be transferable to others.

The Steamer project began in 1979 as an independent research effort that later transitioned to an exploratory development program. The current advanced development effort began about FY 1985.

Previous reports on the Steamer project described an initial framework for developing techniques for automatically generating explanations of how to operate complex physical devices (Stevens & Steinberg, 1981); a user's manual for the Steamer interactive graphics package (Stead, 1981); a method for generating explanations using qualitative simulation (Forbus & Stevens, 1981); CONLAN, a constraint-based programming language for describing the operation of complex devices (Forbus, 1981); a mathematical simulation of the Steamer propulsion plant (Roberts & Forbus, 1981); the then-current Steamer prototype and basic support software (Stevens, Roberts, Stead, Forbus, Steinberg, & Smith, 1982); a computer-based training system for monitoring a boiler light-off procedure (Hutchins, Roe, & Hollan, 1982); and an onsite evaluation of the Steamer prototype at the Navy Surface Warfare Officers School (SWOS) in Newport (Stevens & Hutchins, 1983).

This report describes the Steamer prototype system components and user interface commands. It establishes a starting point for Steamer II, the name applied to recent work on Steamer. It is intended for use by Navy training managers and computer-based training system designers.

B. E. BACON  
Captain, U.S. Navy  
Commanding Officer

J. S. McMICHAEL  
Technical Director

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
NTIS TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Special
A-1	



## SUMMARY

### Problem

A Over the last several years, Navy Personnel Research and Development has produced a prototype simulation of a 1200-psi steam plant. This simulation, called Steamer, is installed on an expensive Symbolics minicomputer at the Surface Warfare Officers School, Pacific (SWOSCOLPAC), Coronado, California. In early 1987, the need to document the Steamer prototype components and the user interface was recognized.

### Objective

The fundamental research goal of the Steamer prototype system was to evaluate the potential of, what was then, new artificial intelligence (AI) hardware and software technology for supporting the construction of computer-based training systems using graphic representations of complex, dynamic systems. The area of propulsion engineering was chosen for a number of reasons.

This ~~technical note~~ describes the Steamer prototype system components and user interface commands and establishes a starting point for designing, developing, and implementing Steamer II.

### Approach

All available documentation about Steamer system was reviewed. The program files on the Symbolics machine were examined to determine how the Steamer prototype software was arranged and operated. All of the menu options presented by the Steamer prototype system were systematically exercised.

### Findings

Careful examination of the actual program code produced an inventory that describes the hardware, system software, application software, and documentation for the Steamer prototype system. Exercising all menu options systematically produced an inventory of all Steamer prototype user interface commands.

### Future Plans

The next step is to create Steamer II by adapting relevant portions of Steamer prototype technology to a microcomputer environment.



## CONTENTS

	Page
INTRODUCTION .....	1
Problem .....	1
Objective .....	1
APPROACH .....	2
DESCRIPTION .....	2
FINDINGS .....	8
DISCUSSION AND FUTURE PLANS .....	8
REFERENCES .....	11
APPENDIX A--STEAMER PROTOTYPE INVENTORY .....	A-0
APPENDIX B--STEAMER PROTOTYPE USER INTERFACE COMMANDS .....	B-0
APPENDIX C--STEAMER PROTOTYPE GLOSSARY .....	C-0
DISTRIBUTION LIST	

## LIST OF FIGURES

1. Graphics editor interface .....	3
2. Steamer control view .....	4
3. Make-up and excess feed view .....	5
4. Steamer prototype system tools .....	7



## INTRODUCTION

### Problem

Over the last several years, Navy Personnel Research and Development Center has produced a prototype simulation of a 1200-psi steam plant. This simulation, called Steamer, is installed on an expensive Symbolics minicomputer<sup>1</sup> at the Surface Warfare Officers School, Pacific (SWOSCOLPAC), Coronado, California. In early 1987, the need to document the Steamer prototype components and the user interface became apparent.

### Objective

The fundamental research goal of the Steamer prototype system was to evaluate the potential of, what was then, new artificial intelligence (AI) hardware and software technology for supporting the construction of computer-based training systems using graphic representations of complex, dynamic systems.

The area of propulsion engineering was chosen for the following reasons:

1. The continuing need to improve Navy training in propulsion engineering has the potential for adequate research funding.
2. Alternative forms of training using simulators or ships are expensive. When the effort began, a high-fidelity simulator, such as the 19E22 steam plant simulator at the SWOSCOLCOM, Newport, would have cost \$7 million. Pierside steaming costs approximately \$6200 per day for a frigate with a 1200-psi steam plant.
3. Access to a detailed mathematical simulation model of a basic dynamic 1200-psi steam plant system enabled efforts to concentrate on the interface, tutorial, and explanation issues.
4. The use of graphical interfaces to support the development of useful mental models of complex dynamic physical systems and devices was of timely interest.
5. Engineering domains seemed to provide the most instructional leverage from the use of AI techniques. Since engineering is concerned with designed systems and physical mechanisms, it appeared to be promising for exploring the nature of mental models.

This technical note describes the Steamer prototype system components and user interface commands and establishes a starting point for designing, developing, and implementing Steamer II.

---

<sup>1</sup>Identification of specific equipment is for documentation only and does not imply any endorsement.



## APPROACH

All available documentation about Steamer system was reviewed. The program files on the Symbolics machine were examined to determine how the Steamer prototype software was arranged and operated. All of the menu options presented by the Steamer prototype system were systematically exercised.

## DESCRIPTION

Navy steam propulsion systems are exceedingly complex physical systems. The propulsion spaces account for about one third of the space in most Navy ships and miles of pipes interconnect thousands of components. The operation of the plant is supervised by an engineering officer of the watch and controlled by a team of 16 to 25 individuals who operate in the most trying of circumstances. They often work long hours in a hot, dirty, and quite dangerous environment. Frequently, individuals must cover more than one watch station in a seemingly unending sequence of watches (6 hours on and 6 hours off). Operators monitor the status of the plant primarily by observing gauges depicting important operational parameters and using other indicators of plant status, particularly how the plant "sounds" and "feels." It takes years of instruction and experience to be able to understand and operate a propulsion plant competently. In addition, rich, robust mental models of the steam plant are needed to be able to respond to the myriad casualty conditions that arise.

The prototype Steamer system consists of a graphical interface to a mathematical simulation model of a steam propulsion plant. The interface enables users to select a specific view from a library of propulsion plant views (Figure 1) and to interact with this view to change the values of the variables in the underlying simulation model. A view is a window into the steam plant. As the model simulates an operating steam propulsion plant, the evolution of plant states, or status, can be observed by graphical changes in the view on a color display.

Views depict aspects of the propulsion system at various levels of detail. They vary from collections of gauges and indicators typically found in a real plant to schematic diagrams that depict conceptualizations similar to those that experts seem to use when thinking about the operation of the propulsion plant. Steamer's increased teaching effectiveness comes from its ability to show: (1) global views of physically dispersed systems in the actual plant that are difficult to see as a total system; (2) simplified versions of systems that are easier to understand or provide better models for reasoning about the plant operation; (3) flow and other internal characteristics of systems or components; and (4) aspects of the system operation that are not normally visible but aid in developing an understanding of that system.

Figures 2 and 3 are black and white renditions of views that users would see on a color screen. State, or status, information is depicted by color, by animation, and by analog, digital, and textual changes. For example, color indicates the operational status of a pump or valve (red for off, green for on), animation shows the flow in pipes, and needles in dials and lines on graphs reflect plant parameters.

Each representation of a component of the plant (e.g., valve, pump, gauge) is called an icon. The iconic representation serves to provide both state information about the components and a mechanism for changing the values of variables in the underlying simulation. Users can change the state of a component with a mouse-controlled cursor by



# Graphics Editor

Create		Attribute	Flavor	Taps	Interact	Activity
Select	Kill	Probe	List	Draw	Initialize	
Save			Reorder	Hardcopy	Configure	
View						
Highlight	Delete	Move	Tap	Draw	Draw	Draw
Clear	Undelete	Copy	Name	Initialize	Show	Show
Mark	Probe	Edit	Color	Configure	Size	Size
All	Default	Shape	Label		Points	Points
Tapped	List	Rotate	Picture		Diagonal	Diagonal
Untapped	Draw	Reflect	Miscellaneous		T Square	T Square
Type	Describe	Inspect				
Find	Inspect					
Misc						
Mark						
Edit Marked Icons						
Circle	Graph	Centrifugal Pump	Bar Switch			
Rectangle	Multi Plot	Rotary Pump	Knife Switch			
Lozenge	Dial	Air Ejector	Rotary Switch			
Triangle	Column	Y Strainer	Toggle Switch			
Trapezoid	Tank	Duplex Strainer	Stop Valve			
Diamond	Digital Bar	Impulse Trap	Anglestop Valve			
Hexagon	Force Bar	Orifice	Check Valve			
Octagon	Bar	Setg	Relief Valve			
Line	Signal	Sadg	Safety Valve			
Spline	Flame	Circuit Breaker	Regulator Valve			
Polygoc	Pipe	Fusible Link	3 Way Valve			
Text		Fuse	4 Way Valve			
Barner		Biscuit	Other			
Icons						
View: Main Engine Lube Oil						
System: Steamer System Model: Steamer Model [stopped] Sub System: All						
Model Running.						

UCSD

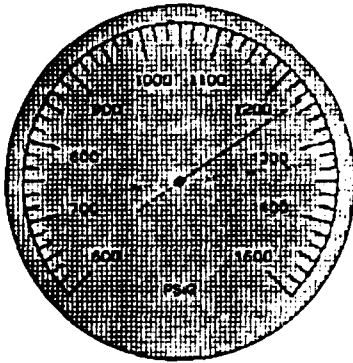
[Fri 27 May 1:04:01] 11:50

Sim: User Input

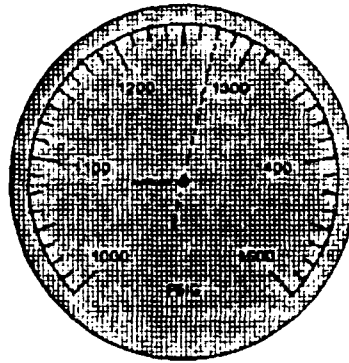
Figure 1. Graphics editor interface.



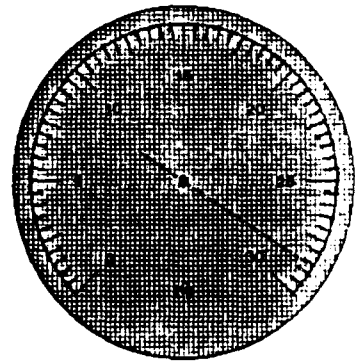
# Steamer Control View



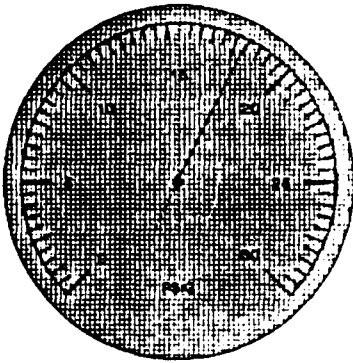
S/H OUTLET



DRUM PRESS



MN COND VACUUM



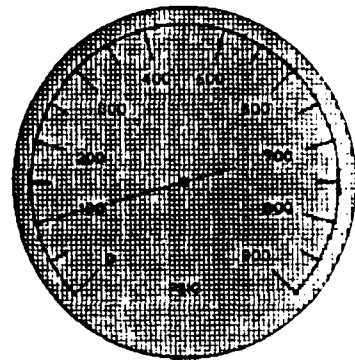
MN LO PRESS

1:5:23

Time

Frozen ☐

Malops ☐



HP 1ST STAGE

15 KTS

125 RPM

1 Boiler

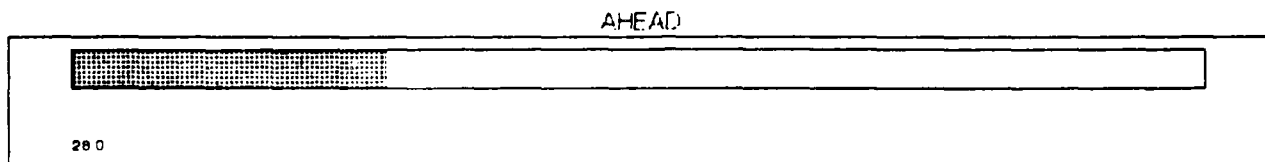


Figure 2. Steamer control view.



# MAKE-UP & EXCESS FEED

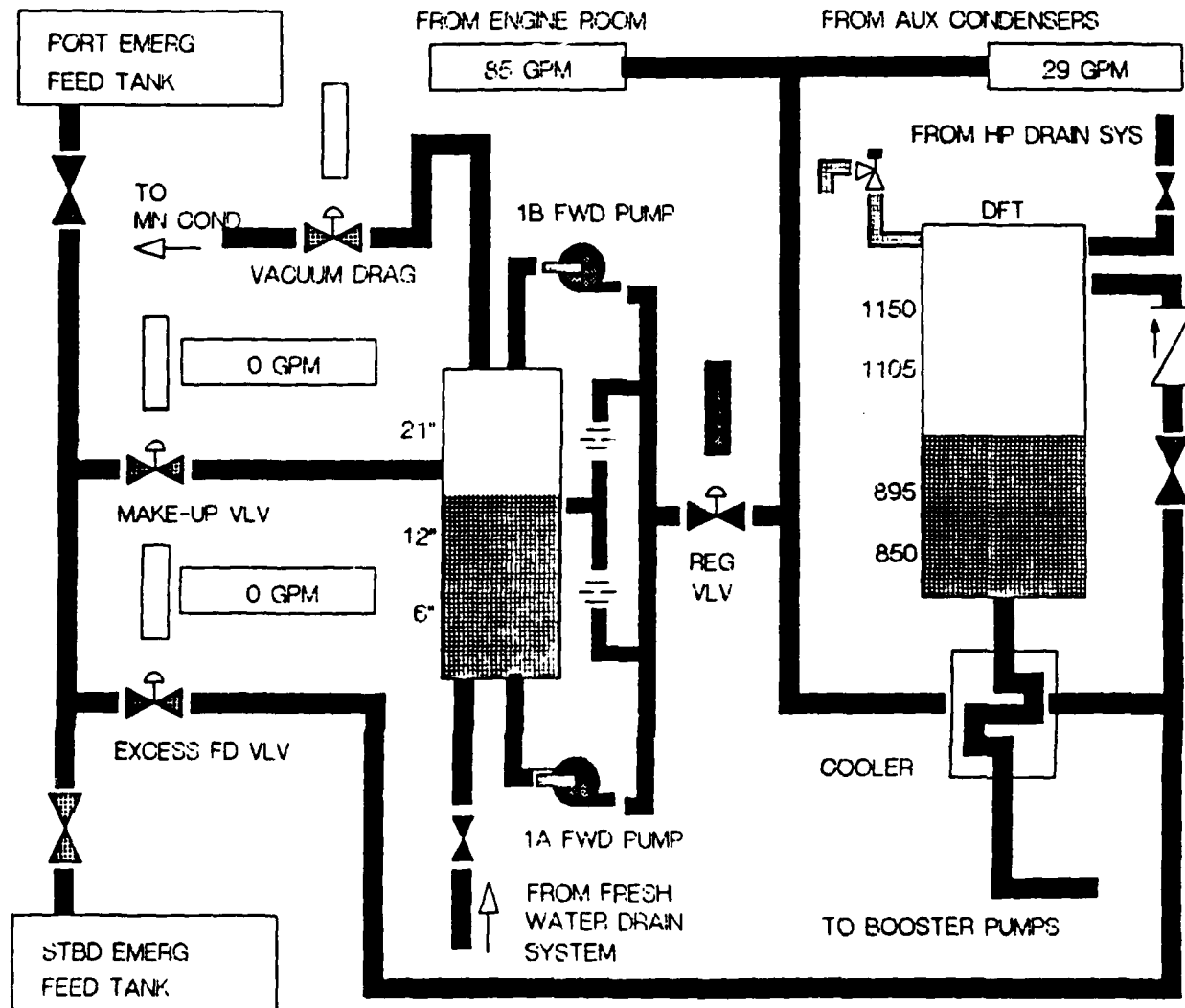


Figure 3. Make-up and excess feed view.



clicking the mouse. For example, clicking the mouse when the cursor is on a pump will change its state (i.e., on, off). Similarly, one can vary the level of a tank, change the value of a dial, or position a throttle. Of course, the nature of the underlying simulation and the goals of the interface designer determine which variables and thus which components can be manipulated. The important point here is that the interface functions as a two-way communication device: depicting and allowing changes of state.

The goal of Steamer prototype system was to build not only a training system specific to the 1200-psi steam plant but also a set of software tools for implementing simulation-based training systems and graphical interfaces for other domains. The following set of tools (Figure 4) evolved as a result of creating the graphical interface. The general simulation environment, called Simenv, which consists of a model controller and a graphics editor, is the core of the Steamer prototype system. It permits one to build interactive graphical interfaces to simulation models or real-time systems. The graphics editor makes available a set of icons, facilities for modifying characteristics of icons (e.g., size, shape, color, and placement), and the ability to associate icons with model variables so that the icons reflect the values of the variables and so that one can interact with the icons to change the values of their associated variables. The model controller enables one to run simulation models, introduce predetermined casualties into the chosen model, observe the model's state via graphical views constructed with the graphics editor, and interact with the views to change the state of a simulation model.

Figure 4 includes several tools whose development was not completed. Chief among them is the icon editor, which would enable users to construct new icons without requiring them to operate at the level of code writing. In addition, a series of knowledge-based editors for the specification of domain knowledge was under development. The behavior editor was supposed to explore the incorporation of simulation knowledge into icons. The lesson editor was intended to explore the incorporation of domain knowledge into graphical views so that they could explain themselves, pose problems to students, and monitor their answers. Designer, an interactive visual design consultant for users of the graphics editor, was supposed to make available graphical design knowledge during the process of constructing and critiquing graphical views.

The mathematical simulation model of the steam plant was adapted from the model developed for use with the 19E22 steam plant simulator at SWOSCOLCOM. The program, originally coded in FORTRAN, was rewritten in ZetaLisp, the version of LISP that runs on the LISP machines built by Symbolics, Inc.

From the beginning, the target computer hardware for the Steamer effort was a low-cost portable environment (Stevens & Steinberg, 1981). Development work on the Steamer prototype system, however, took place on larger machines. The development environment provided by the larger machines contributed significantly to the iterative design and development of the Steamer prototype system.

The Steamer prototype system resides on a single-station minicomputer made by Symbolics. Each computer costs approximately \$85,000 to buy and approximately \$15,000 per year to maintain. Both costs are likely to increase in future years. The central processing unit (CPU) is the size of a two-drawer filing cabinet and exceedingly noisy when operating.



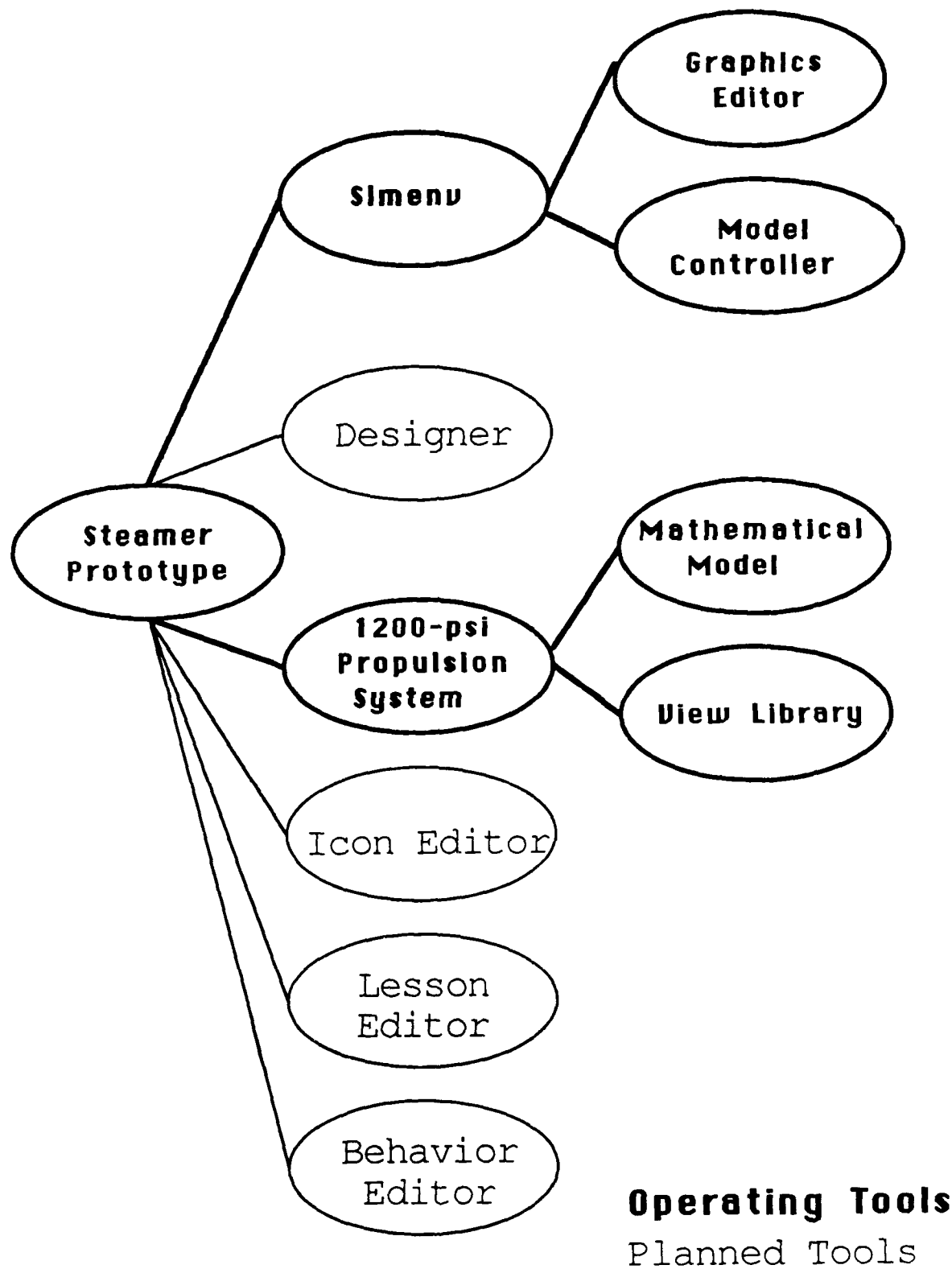


Figure 4. Steamer prototype system tools.



SWOSCOLPAC uses one of these computers for the Steamer prototype system. The CPU is in a remote computer room. The two terminals used, the monochrome console and the color display, are in another room. They are not in the classroom used for 1200-psi steam plant training because it is too far from the CPU. The fact that only one student can use the system at a time also limits use of the Steamer prototype system.

The SWOSCOLPAC personnel have indicated that the basic features of the Steamer prototype system are desirable but that its present state limits its use. They use the Steamer prototype system only when a student has trouble understanding a particular aspect of the steam plant operation or wants to use it after normal class hours. The prototype has not been incorporated into the course curriculum.

Ideally, with a Steamer system on each student's desk, instructors could guide the class through planned casualty situations while the students observe the results of these casualties on the plant. Instructors could also prepare and store structured lessons for use by students in every class. Students having trouble in specific areas could be directed to relevant additional stored lessons.

## FINDINGS

Much of the available documentation about the Steamer prototype system concentrated on the theoretical problems of teaching the operation of large-scale physical devices using computer-based instruction techniques. Teaching thus provided would substitute for practicing on the physical devices themselves.

The Steamer prototype inventory (Appendix A) describes the hardware, system software, and application software and lists all documentation for the Steamer prototype system. It does not discuss the capabilities, effectiveness, or usefulness of the components.

Systematic exercising of all menu options produced the inventory of Steamer prototype commands (Appendix B). This inventory describes the Steamer prototype system user interface commands.

The Steamer prototype glossary in Appendix C defines the terminology specific to Steamer.

## DISCUSSION AND FUTURE PLANS

The Steamer prototype system determined the feasibility of using interactive high-resolution color graphics to depict the dynamics of an extremely complicated system. In conjunction with Steamer's mathematical simulation model, which represents a verified data base of domain information, the resulting simulation has proved very effective in training. Even without supporting quantifiable data, there is enough support for the concepts behind the prototype Steamer to justify the development of fleet training based on the Steamer prototype system (Stevens & Hutchins, 1983).



Current efforts and plans include an in situ assessment of Steamer II, an assessment of the Steamer II system in a variety of potential operational applications, and collecting descriptive data on the use of the system, performance data from students, and data on instructor, student, and staff experiences and opinions about Steamer II.

Preliminary investigations have shown that incorporating video images into Steamer II may be useful. Therefore, the feasibility and cost effectiveness of integrating the videodisc capability with the graphics editor will be investigated. This capability will enable video images of propulsion plant components and operational procedures to be displayed. These images will be merged with graphical representations.

A lesson editor will be developed to allow instructors to create automated tutorials. These tutorials will include diagrams that can explain what they depict, problems that students will answer by interacting with the diagrams, and the ability to monitor student interactions.

Moving Steamer to a more compact, reliable, and relatively inexpensive computer system is underway. Computer hardware has advanced to the point where a system such as Steamer can potentially run on a relatively inexpensive and compact microcomputer. Currently available computers under investigation include those that are based on Intel 80386 or Motorola 68020 processors. The future promises even more speed in less physical space. We intend to develop an easily portable system that can be distributed to all sites where steam plant operation is part of the curriculum. In addition, SWOSCOLPAC personnel have indicated that a similar on-board training capability would be extremely valuable. A compact system on each ship with a 1200-psi steam plant would be a logical outcome.

Specification of Steamer II delivery systems is being done with an eye toward the next generation of improvements and additions that will result in a powerful and broadly useful training/simulation system. Other mathematical simulation models around which a similar graphically-based simulation can be built using the same set of tools available in Steamer II are being researched. Different domain data bases are being examined, such as a manpower management training simulator called \*IMAGE\*, which used the graphics editor in its development at Navy Personnel Research and Development Center. Expert system shells will be investigated as potential substitutes for a mathematical simulation model and as the basis for training/simulation using the same tools. Intelligent tutors and adaptive instructor/student models will be researched. A generalizable "simulation editor" is a possibility.



## REFERENCES

- Forbus, K. D. (1981, August). Project STEAMER: IV. A primer on CONLAN--A constraint-based language for describing the operation of complex physical devices (NPRDC TN 81-26). San Diego: Navy Personnel Research and Development Center.
- Forbus, K. D., & Stevens, A. (1981, August). Project STEAMER: III. Using qualitative simulation to generate explanations of how to operate complex physical devices (NPRDC TN 81-25). San Diego: Navy Personnel Research and Development Center.
- Hutchins, E., Roe, T., & Hollan, J. (1982, August). Project STEAMER: VII. A computer-based system for monitoring the boiler light-off procedure for a 1078-class frigate (NPRDC TN 82-25). San Diego: Navy Personnel Research and Development Center.
- Roberts, B., & Forbus, K. (1981, August). Project STEAMER: V: Mathematical simulation of STEAMER propulsion plant (NPRDC TN 81-27). San Diego: Navy Personnel Research and Development Center.
- Stead, L. (1981, August). Project STEAMER: II. User's manual for the STEAMER interactive graphics package (NPRDC TN 81-22). San Diego: Navy Personnel Research and Development Center.
- Stevens, A., & Hutchins, E. (1983, September). Project STEAMER: VIII. System evaluation by Navy propulsion engineering training personnel (NPRDC SR 83-52). San Diego: Navy Personnel Research and Development Center. (AD-A134 051)
- Stevens, A., Roberts, B., Stead, L., Forbus, K., Steinberg, C., & Smith, B. (1982, January). Project STEAMER: VI. Advanced computer-aided instruction in propulsion engineering--an interim report (NPRDC TR 82-28). San Diego: Navy Personnel Research and Development Center. (AD-A110 797)
- Stevens, A., & Steinberg, C. (1981, August). Project STEAMER: I. Taxonomy for generating explanations of how to operate complex physical devices (NPRDC TN 81-21). San Diego: Navy Personnel Research and Development Center.



**APPENDIX A**  
**STEAMER PROTOTYPE INVENTORY**



The simulation environment (often referred to as Simenv) is the core of the Steamer prototype system. It provides the tools for producing a simulation. The graphics editor creates views of the system and the model control facility is used for interacting with a simulation.

The Steamer prototype simulation was produced using the Symbolics editor (for creating ZetaLisp code) and the simulation environment (for building the simulation). Any simulation that is produced using the simulation environment is composed of two parts: (1) the mathematical simulation model, a mathematical representation of some physical system, which in the case of Steamer, is the physical steam plant and (2) the library of views. Each view is a diagram of a portion of the physical plant and is composed of a set of graphic icons. Views are created using the graphics editor.

Code was found for the lesson editor, icon editor, and designer. None of these processes is operational in the Steamer prototype system.

The remainder of Appendix A lists the inventoried components of the Steamer prototype system. The components are grouped into the categories of hardware, system software, application software, and documentation.

## HARDWARE

The minimal hardware on which the Steamer prototype system runs follows:

1. Symbolics 3640 Computer containing:
  - 2560K words of memory (36 bit words)
  - 2 hard disks, each with 110 Megawords of storage
  - Streaming tape drive
2. Symbolics monochrome monitor with keyboard (1150 x 900 resolution).
3. Symbolics color monitor with 256 colors (1280 x 1024 resolution).
4. Logitech mouse.

The Steamer prototype system software has also been run on other Symbolics 3600-series computers (3600, 3670, 3675).



## SYSTEM SOFTWARE

Because the Symbolics machine is a LISP machine, the operating system, programming language, and hardware are all very closely integrated. There is not a distinct boundary between the operating system and the LISP interpreter/compiler.

The complete set of system software used for implementing Steamer prototype is as follows:

1. Genera 7.1 Operating System.
2. Symbolics Color System--version 331.5. This includes Symbolics Color Support--version 11.4.
3. ZetaLisp Programming Language. ZetaLisp can be either interpreted or compiled.
4. Flavors extension for ZetaLisp for providing object-oriented programming.

## APPLICATION SOFTWARE

The file system subdirectories that exist on the Symbolics machine and the LISP files they contain are documented below. In addition, another set of files is used to load and configure each of the processes on the Symbolics. These files are scattered throughout the file directories, but for the purposes of this document they will be grouped together.

Each of the Steamer prototype processes is composed of one or more packages that is loaded onto the Symbolics machine. Each package is identified by a title that is stored in the configuration files. The titles for each of the three primary subsystems are listed below:

1. Simulation Environment
2. Steam Plant Simulation
  - Steamer System
  - Steamer System Views
  - Steamer System Model
3. Lesson Editor
  - Experimental Lesson Editor
  - Steamer System Lessons
  - Steamer Model



## APPLICATION SOFTWARE

### Configuration Files

Listed below are the configuration files used to load each of the processes. These files provide information to the operating system about the content of the processes and the configurations needed for the processes to run.

Configuration files for the simulation environment:

1. defsimenv.lisp
2. simenv.system
3. simenv.translations
4. simenv-documentation.system

Configuration files for the steam plant simulation:

1. defsteamer-system.lisp
2. steamer-system.system
3. steamer-system.translations
4. steamer-model.system
5. steamer-model.translations

Configuration files for the lesson editor:

1. deflesson.lisp
2. lesson.system
3. lesson.translations

Configuration files for the icon editor:

1. defie.lisp

Configuration files for designer:

1. defdesigner.lisp
2. defdesigner-knowledge.lisp
3. defdesigner-system.lisp



## **APPLICATION SOFTWARE**

### **Simulation Environment**

This Symbolics directory contains the code for the simulation environment.

#### **model subdirectory**

This subdirectory contains the high level simulation control routines. These routines interpret the operator interactions and call lower level routines for performing the actions. Simulation control is often referred to as model control, hence the name model for this subdirectory.

1. model-activity
2. model-defs
3. model-functions
4. model-panes

#### **system subdirectory**

Listed below are the low level simulation control routines that perform the majority of the simulator functions.

- |                          |                         |
|--------------------------|-------------------------|
| 1. activities            | 14. segment-support     |
| 2. activity-control      | 15. sim-initializations |
| 3. basic-lesson          | 16. simenv-activity     |
| 4. basic-model           | 17. simenv-commands     |
| 5. basic-view            | 18. simenv-defs         |
| 6. file-control          | 19. simenv-frame        |
| 7. graphic-pane-activity | 20. simenv-panes        |
| 8. lc-mixin              | 21. simenv-control      |
| 9. mc-mixin              | 22. status-defs         |
| 10. package              | 23. status-macros       |
| 11. pane-defs            | 24. system              |
| 12. save-mixin           | 25. title-activity      |
| 13. segment-defs         | 26. vc-mixin            |
| 14. segment-support      | 27. version-control     |
| 15. sim-initializations  | 28. view-set            |

#### **editor subdirectory**

Listed below are the graphics editor routines that are used to create and modify views of a simulation.

1. editor-activity
2. editor-defs
3. editor-edit
4. editor-grid
5. editor-hardcopy
6. editor-mark
7. editor-mouse-icons
8. editor-panes
9. editor



## icons subdirectory

Each simulation view is composed of a group of graphical primitives called icons. The graphics editor provides a standard set of icons that includes such things as pipes, valves, pumps, tanks, electrical symbols, and graphs. The icons subdirectory contains the code for these standard icons.

- |                 |                      |
|-----------------|----------------------|
| 1. 3-way-valve  | 15. graph            |
| 2. 4-way-valve  | 16. line             |
| 3. arc          | 17. multi-plot-graph |
| 4. bar          | 18. old-valve        |
| 5. beam-tot     | 19. pipe             |
| 6. column       | 20. polygon          |
| 7. data-set     | 21. pump             |
| 8. devices      | 22. region           |
| 9. dial         | 23. signal           |
| 10. digital-bar | 24. spline           |
| 11. electric    | 25. switch           |
| 12. flame       | 26. text             |
| 13. force-bar   | 27. valve            |
| 14. geometric   | 28. view-icon        |

## icon-support subdirectory

This subdirectory contains general icon-building subroutines and flavors. These subroutines are used by the standard icons stored in the icon subdirectory. They can also be used by ZetaLisp programmers to build other icons.

- |                      |                    |
|----------------------|--------------------|
| 1. action            | 15. icon-defs      |
| 2. aspect-ratio      | 16. icon-m         |
| 3. basic             | 17. map            |
| 4. continuous        | 18. no-tap-mapping |
| 5. defmaps           | 19. picture-mixin  |
| 6. discrete-color    | 20. point          |
| 7. discrete          | 21. points         |
| 8. display           | 22. position       |
| 9. draw-self         | 23. rectangular    |
| 10. element-position | 24. rotation       |
| 11. element          | 25. setup          |
| 12. fill             | 26. show           |
| 13. gage             | 27. tap            |
| 14. icon-animation   |                    |



**graphics subdirectory**

Listed below are the low level graphics routines for drawing and positioning objects on the screen.

- |                       |                     |
|-----------------------|---------------------|
| 1. 7-0-color-patch    | 12. gl-m            |
| 2. aed-m              | 13. gl-text         |
| 3. aed                | 14. gl              |
| 4. b&w                | 15. graphics-defs   |
| 5. beep               | 16. pipe-blinker    |
| 6. device-animation   | 17. pipe-object     |
| 7. gi                 | 18. pipe92          |
| 8. gl-devices         | 19. postscript-view |
| 9. gl-initialize-aed  | 20. postscript      |
| 10. gl-initialize-b&w | 21. sc-m            |
| 11. gl-initialize-sc  | 22. sc              |

**utilities subdirectory**

The utility routines below provide tools for performing functions such as building menus, manipulating strings, and performing calculations.

1. font-defs
2. format-mouse-patch
3. format-mouse
4. hcm
5. list
6. menu-choose-utilities
7. numeric
8. overlap
9. sct-modules
10. string
11. turbo-menu

**site subdirectory**

The site directory contains routines for configuring the simulator to a specific computer.

1. site-configuration-data
2. site-configuration



**videodisc subdirectory**

This subdirectory contains programs for interfacing to a videodisc player. Since no videodisc was connected to the system, it is not known how well this interface works, if at all.

1. discovision
2. vd-initialize-discovision
3. videodisk

**documentation subdirectory**

The documentation subdirectory contains a set of files that briefly explain the operation of the simulation environment. These files are discussed in detail later.

**Steam Plant Simulation**

This Symbolics directory contains the code for the simulation of the 1200-psi steam plant. The main steam plant simulation directory contains one file, steamer-system.lisp. This file defines the global parameters for the steam plant simulation. It is entirely machine generated using the simulation environment.

**models subdirectory**

These routines comprise the underlying mathematical simulation model for the 1200-psi steam plant.

- |                     |                          |
|---------------------|--------------------------|
| 1. auxiliary        | 10. ic                   |
| 2. casualties       | 11. initializations      |
| 3. casualty-defs    | 12. patch                |
| 4. commands         | 13. run-defs             |
| 5. common           | 14. run                  |
| 6. defsteamer-model | 15. steamer-model        |
| 7. engineroom       | 16. steamer-package-defs |
| 8. fireroom         | 17. wpump                |
| 9. fortran          |                          |



**models >augments subdirectory**

These files augment the mathematical model. Augments calculate additional steam plant values from the basic mathematical model. When views are built using the simulation environment, these additional values can be displayed. For example, the rate of flow in a pipe can be calculated as an augment based on the speed and capacity of the pump that fills the pipe. The rate of flow can then be displayed in a view that contains the pipe.

1. abc-water-level-control-model-augment
2. ae-sys-model-augment
3. as-1200-sys-model-augment
4. as-150-sys-model-augment
5. as-superheater-protection-model-augment
6. augment-support
7. fo-burner-front-lb-model-augment
8. fo-control-valve-lb-model-augment
9. fo-sys-basic-model-augment
10. fo-sys-model-augment
11. gs-er-model-augment
12. gs-unloader-er-model-augment
13. lo-sys-model-augment
14. make-up-excess-feed-model-augment
15. mc-air-ejector-model-augment
16. mc-pump-submergence-model-augment
17. mc-sys-model-augment
18. mf-control-model-augment
19. mf-sys-model-augment
20. ml-jet-pumps-model-augment
21. msw-circ-model-augment



## views subdirectory

Listed below are the files containing the code for the views for the 1200-psi steam plant. Each view corresponds to a single graphical picture of a portion of the steam plant. The views are created using the graphics editor portion of the simulation environment. Figures 2 and 3 (text) are are samples of views created by the graphics editor. Each of these files is entirely machine generated.

1. 1200-psi-aux-steam
2. 150psi-auxiliary-steam
3. 1b-burner-front
4. abc-1b-boiler
5. abc-big-picture
6. abc-schematic
7. ae-lamr
8. autosignal-flow-1b
9. aux-exhaust-augmentor
10. auxiliary-exhaust
11. auxiliary-exhaust-fr
12. auxiliary-steam-lamr
13. basic-fo-system
14. basic-lube-oil-system
15. control
16. electric-1
17. fo-control-valve-1b
18. fuel-oil-service-pump-1a
19. fuel-oil-system
20. gland-seal-er
21. gland-seal-unloader
22. jet-pump-lab
23. lube-oil-unloader
24. main-circ-system-er
25. main-condensate-er
26. main-engine-lube-oil
27. main-feed-control
28. main-feed-system
29. mc-air-ejector
30. mc-pump-submergence
31. mfp-lube-oil-cooler
32. mfp-recirc-sys-1a
33. superheater-protection
34. throttleboard
35. water-level-control



**views >sc subdirectory**

These are additional views of the 1200-psi steam plant.

1. console-lb
2. make-up-and-excess-feed
3. steam-cycle

**lessons subdirectory**

This subdirectory contains lessons that were built using the lesson editor. Even though these files are loaded as part of the lesson editor subsystem, they are stored as part of the steam plant simulation subsystem.

1. boiler-control
2. muef-lesson-1

**Lesson Editor**

The lesson editor was intended to be used to build sequenced lessons that are based on an existing simulation. Two lessons built for the 1200-psi steam plant simulation are stored in the lessons subdirectory for the simulation. This directory contains the code for the lesson editor and has no subdirectories. Only a portion of the lesson editor is operational.

1. highlighter
2. lesson-activity
3. lesson-defs
4. lesson-edit
5. lesson-functions
6. lesson-init
7. lesson-mouse
8. lesson-pane-defs
9. lesson-panes
10. lesson-status-defs
11. package
12. scroll-region



## **APPLICATION SOFTWARE**

### **Icon Editor**

#### **activity subdirectory**

1. discrete-color
2. fill
3. i.e-activity
4. i.e-commands
5. i.e-defs
6. i.e-init
7. i.e-pane-defs
8. i.e-panes
9. i.e-status
10. i.e-status-functions
11. package
12. position

#### **activity>draw subdirectory**

1. action-setup
2. draw-primitives
3. fixed-color-mapping
4. line
5. new-bar
6. oogs

#### **activity>i.e-patch>i.e-7 subdirectory**

1. frame
2. i.e-activity
3. i.e-init
4. i.e-pane-defs
5. i.e-panes
6. package
7. windows

#### **new-icons subdirectory**

1. composite-icon
2. dial
3. multi
4. new-bar
5. ornament



## **APPLICATION SOFTWARE**

### **Designer**

#### **activity subdirectory**

1. basic-view-critique
2. designer-activity
3. designer-pane-defs

#### **atms subdirectory**

1. tms-interface

#### **class-defs subdirectory**

1. basic-graphics
2. constraint-defs
3. graphic-type-defs
4. principle-defs
5. standard-defs
6. technique-defs

#### **constraints subdirectory**

1. constraints
2. principles
3. standard

#### **critique subdirectory**

1. critique
2. critique-comment
3. critiquee

#### **graphics subdirectory**

1. designer-sc

#### **objects subdirectory**

1. certainty
2. elements
3. icon-interface



## **APPLICATION SOFTWARE**

**Designer**

### **principles subdirectory**

1. significant-difference

### **relations subdirectory**

1. basic-layout
2. class
3. grouping
4. proximity
5. relations
6. repetition
7. similarity

### **standards subdirectory**

1. pipe-size
2. titles

### **style subdirectory**

1. style

### **synthesis subdirectory**

1. synthesis

### **top level subdirectory**

1. designer-globals
2. designer-m
3. package
4. utilities

### **designer-system subdirectory**

1. designer-system



## DOCUMENTATION

There is a large variety of documentation available for the prototype Steamer system. Unfortunately, most of it is old or is very general in its description of the system. The documentation includes operations manuals for the Symbolics machine, reports outlining research work on the Steamer project, descriptions of the steam plant mathematical model, text files describing the operation of the simulation environment, and other miscellaneous documentation. This section of Appendix A lists all of the documentation that is known at this time.

### Simulation Environment

The documentation for the simulation environment consists of a set of text files that are stored in the documentation subdirectory of the simulation environment and listed below. The files provide a general description of the commands that are available when using the simulation environment. Using the commands, an author can create a simulation that contains the following elements: a single system, one or more subsystems, a model, and a number of views.

1. `overview.text`--presents an overview of the simulation environment and describes the format of the screen, the operation of the mouse, and the available commands.
2. `creating-system.text`--describes the commands for creating and modifying systems.
3. `creating-sub-systems.text`--describes the commands for creating and modifying subsystems.
4. `creating-models.text`--describes the commands for creating and modifying models.
5. `creating-views.text`--describes the commands for creating views of a model.
6. `distributing-simenv-in-7-0.text`--describes how to produce distribution tapes for the simulation environment.
7. `simenv-readme.text`--describes the steps for loading the simulation environment and the steam plant simulation from tape and how to get it running.



## DOCUMENTATION

### Steamer

The Steamer reports describe some of the research that led to the development of the Steamer prototype system. These descriptions do not necessarily reflect the current software.

1. Stevens, A., & Steinberg, C. (1981, August). Project STEAMER: I. Taxonomy for generating explanations of how to operate complex physical devices (NPRDC TN 81-21). San Diego: Navy Personnel Research and Development Center.
2. Stead, L. (1981, August). Project STEAMER: II. User's manual for the STEAMER interactive graphics package (NPRDC TN 81-22). San Diego: Navy Personnel Research and Development Center.
3. Forbus, K. D., & Stevens, A. (1981, August). Project STEAMER: III. Using qualitative simulation to generate explanations of how to operate complex physical devices (NPRDC TN 81-25). San Diego: Navy Personnel Research and Development Center.
4. Forbus, K. D. (1981, August). Project STEAMER: IV. A primer on CONLAN--A constraint-based language for describing the operation of complex physical devices (NPRDC TN 81-26). San Diego: Navy Personnel Research and Development Center.
5. Roberts, B., & Forbus, K. (1981, August). Project STEAMER: V: Mathematical simulation of STEAMER propulsion plant (NPRDC TN 81-27). San Diego: Navy Personnel Research and Development Center.
6. Stevens, A., Roberts, B., Stead, L., Forbus, K., Steinberg, C., & Smith, B. (1982, January). Project STEAMER: VI. Advanced computer-aided instruction in propulsion engineering--an interim report (NPRDC TR 82-28). San Diego: Navy Personnel Research and Development Center. (AD-A110 797)
7. Hutchins, E., Roe, T., & Hollan, J. (1982, August). Project STEAMER: VII. A computer-based system for monitoring the boiler light-off procedure for a 1078-class frigate (NPRDC TN 82-25). San Diego: Navy Personnel Research and Development Center.
8. Stevens, A., & Hutchins, E. (1983, September). Project STEAMER: VIII. System evaluation by Navy propulsion engineering training personnel (NPRDC SR 83-52). San Diego: Navy Personnel Research and Development Center. (AD-A134 051)
9. Hollan, J., Hutchins, E. L., & Weitzman, L. M. (1984). Steamer: An interactive inspectable simulation-based training system. AI Magazine, 5(2), 11-28. This article, which is probably the best general description of the Steamer system and its capabilities, was written by some of Steamer's primary researchers.



## DOCUMENTATION

### View Descriptions

These files describe the components for some of the 1200-psi steam plant views. Descriptions are available for the following views:

1. 1200 PSI Auxiliary Steam
2. 150 PSI Auxiliary Steam
3. Auxiliary Exhaust Augmentor
4. Auxiliary Exhaust
5. Basic Fuel Oil System
6. Basic Lube Oil System
7. Console 1b
8. Control
9. Fuel Oil Control Valve 1b
10. Fuel Oil Service Pump
11. Gland Seal Engineroom
12. Gland Seal Unloader
13. Main Circulation System Engineroom
14. Main Condensate Engineroom
15. Main Engine Lube Oil
16. Main Feed Control
17. Make-up and Excess Feed
18. MC Air Ejector
19. Main Feed Pump Recirculation System 1a
20. Steam Cycle
21. Superheater Protection

### Miscellaneous

The following documents were used for this effort, but are not readily available.

1. Gould, Inc. produced a five-volume report entitled, Final Math Model Report for the 1200 PSI Propulsion Plant Trainer--May 31, 1978. This report describes the underlying mathematical model for the steam plant simulation. It was originally produced as support documentation for the 19E22 steam plant simulator at Surface Warfare Officers School Command, Newport.
2. A layman's Guide to Steamer is a basic operator's manual for running the simulation environment and the steam plant simulation. It does not fully describe all of the features, but does provide a good introduction for getting around in the system.



**APPENDIX B**  
**STEAMER PROTOTYPE USER INTERFACE COMMANDS**



This Appendix emphasizes the processes that are installed and operating. The operating processes are Simenv (consisting of the model control facility and the graphics editor) and the lesson editor, which is only partially operational. No attempt was made to document the operation of other processes, since they are not operable.

To understand the operation of the Steamer prototype system, it is helpful to understand its four major functions: (1) building a simulation, (2) displaying a simulation, (3) building a lesson, and (4) displaying a lesson. Simenv provides the commands for building and displaying simulations, while the lesson editor provides the commands for building and displaying lessons.

The commands are listed in a sequence that reflect their grouping in the user interface. Steamer has three user interface formats. They are model control, graphics editor, and lesson editor. Each format is composed of a number of windows, and each window contains a set of text icons. Each text icon corresponds to a Steamer prototype command, and the text icons in a particular window usually perform related functions. The operator selects a specific command by using the mouse to move the cursor over a text icon. Once the cursor is positioned over an icon, the command is activated by clicking one of the three mouse buttons. In many cases, each mouse button causes a *different command* to be activated. In addition, a status line provides additional icons and additional commands for the operator. The way the user interface is organized causes many commands to be repeated two or three times throughout the system. The organization of the user interface is as follows:

1. Model Control
  - Model commands
  - View commands
  - Simenv commands
  - Status line commands
2. Graphics Editor
  - View commands
  - Simenv commands
  - Mark commands
  - Edit marked icon commands
  - Grid commands
  - Icon selection
  - Status line commands
3. Lesson Editor
  - Lesson commands
  - View commands
  - Simenv commands
  - Segment commands
  - Edit commands
  - Status line commands
  - Lesson status commands
4. Operating system commands

See Appendix C for a glossary of Steamer terminology.



## MODEL CONTROL

Model Control (Figure B-1) is used to control a model, select views, and interact with the views.

### Model Commands

See Figure B-1, top left window.

**Run**--Starts execution of the model. Once started, the model will continue to execute until stopped.

**Stop**--Stops execution of the model.

**Reset**--Resets the model to an initial state. This is done by calling a reset routine that is defined as part of the model. There are four possible reset states for the Steamer system:

- Cold iron
- Auxiliary steaming
- Underway (15 knots, one boiler)
- Underway (25 knots, two boilers)

**Rate**--Sets the frequency of execution for the model. If the rate is set at five seconds, the model is updated (executed) once every five seconds. The default rate is one second.

**Tick**--Executes the model once. The model is run for one tick.

**Status**--Shows the current status of the model. This is done by calling a status routine that is defined as part of the model.

**Save**--Acts according to which mouse button is pushed:

Save the Current Model--Saves the currently selected model on the disk.

Save all System Models--Saves all models for the current system on the disk.

**Find**--Finds and loads a model from a disk file.

**Select**--Selects a loaded model.

**Misc**--Performs miscellaneous model control functions. This command is implemented by calling a model control routine that is defined as part of the model. The following miscellaneous functions are defined for the Steamer prototype system:

**Set Casualties**--Allows the user to activate predefined states within the model. For the Steamer prototype system, these casualties are as follows:

- Fireroom--25 fireroom casualty states
- Engineroom--23 engineroom casualty states
- Auxiliary I and II--25 auxiliary casualty states
- Electrical Central--25 electrical casualty states
- All--Composed of all of the above



Model Control				
Run Stop Reset Model	Rate Tick Status	Save Find Select	Misc	Activity
View		Select Interact Probe	Initialize Configure Lesson	Simenv
<div style="display: flex; justify-content: space-between;"> <div>Control View</div> <div>View: Main Engine Lube Oil</div> </div>				
<div style="display: flex; justify-content: space-between;"> <div> <i>Lisp</i>            System: Steamer System    Model: Steamer Model [stopped] Sub System: All         </div> <div> <i>mpc</i>            UCSO         </div> </div>				

[Fri 27 May 1:04:23] 11:50pm  
 User Input

Figure B-1. Model control interface.



## MODEL CONTROL

## Model Commands

Reset Casualties--Allows the user to reset casualties that were previously set.

Show Malops--Shows current malfunctioning operations within the model.

## View Commands

See Figure B-1. top middle window.

Select--Acts according to which mouse button is pushed:

- Select a view--Selects a defined view and displays it either on the color monitor or in the left display box on the monochrome monitor. This view is called the normal view.
- Select a control view--Selects a defined view and displays it in the right display box on the monochrome monitor.

Interact--Interacts with the current view. The cursor is moved to the current view on the color monitor (or the left portion of the monochrome monitor), and the operator clicks the mouse while the cursor is positioned over a graphic icon. If the icon has an output tap (set tap) into the model, the state variable associated with the icon is changed.

Probe--Updates the icon in the currently displayed views with the values from the associated state variables in the model.

Initialize--Acts according to which mouse button is pushed:

- Initialize Icon Graphics System
- Initialize System Model Process

Configure--Displays a menu of configurations:

- Normal--Causes normal views to be displayed on the color monitor.
- B&W--Causes normal views to be displayed in the left display box on the monochrome monitor.

Lesson--Activates a defined lesson. For the Steamer prototype system, two lessons are defined:

- Boiler Control
- Muef Lesson 1

## Simenv Commands

See Figure B-1, top right window.

Activity--Allows selection of different processes:

- Graphics Editor
- Lesson Editor



## **MODEL CONTROL**

### **Status Line Commands**

These commands are chosen by positioning the cursor over one of the selections on the status line and clicking one of the mouse buttons. The status line selection consists of category headings (system, model, subsystem, and views) and the currently selected item for each category (for example, Steamer system is the selected system). The current state of the model (running/stopped) is also shown on the status line.

### **system commands**

See Figure B-1, lower left: System: Steamer system.

Select a System--Selects a loaded system.

Create a New System--Creates a new system. When a new system is created, the user must specify the following information:

- System name
- System physical path

Find a System--Finds and loads a system from disk.

### **system item commands**

Change System Attributes--Allows changes to defined attributes for the selected system. The following attributes can be changed:

- System name
- Whether files should be compiled after saves
- System version
- Model version
- View version
- Lesson version
- System package files
- System physical path
- Model physical path
- View physical path
- Lesson physical path

Save System--Saves the current definition of the selected system.

Kill System--Removes a system. This is the opposite of loading a system.

Probe System--Causes the displayed views to be updated with values from the associated state variables in the model.



**model commands**

Select a Model--Selects an existing model.

Create a New Model--Creates a new model. When a new model is created, the user must specify the following information:

- Model name
- Model routine that performs the run command
- Model routine that performs the reset command
- Model routine that performs the reset state menu command
- Model routine that performs the status command
- Model routine that performs miscellaneous commands
- A list of auxiliary files
- The system name for making the system
- Model physical path

Find Model--Finds and loads a model from the disk.

Save All Modified Models--Saves all models that have been updated.

Save All Models--Saves all models.

Display Model States--Displays the current state of all known models, indicating whether the model is loaded and whether it has been modified since being loaded.

**model item commands**

Change Model Attributes--Changes the attributes for the current model. The following attributes can be changed:

- Model name
- Model routine that performs the run command
- Model routine that performs the reset command
- Model routine that performs the reset state menu command
- Model routine that performs the status command
- Model routine that performs miscellaneous commands
- A list of auxiliary files
- The system name for making the system
- Model physical path

Save Model--Saves the current model.

Kill Model--Removes the current model. This is the opposite of adding a model.

Reset Model--Resets the model to an initial state.

Model Status--Displays the status of the current model.

Model Misc Functions--Allows selection of miscellaneous model control functions.

Model Item State Commands--The current state of the model is displayed as "stopped" or "running."



**subsystem commands**

Select a subsystem--Selects an existing subsystem.

Create a New Subsystem--Creates a new subsystem. When a new subsystem is created, the user must specify subsystem name.

**subsystem item commands**

Change Subsystem Attributes--Changes the attributes for the current subsystem.

Edit Subsystem Views--Allows the user to specify which views are to be included in the current subsystem.

Kill Subsystem--Removes a subsystem. This is the opposite of creating a subsystem.

**view commands**

Select a View--Selects an existing view. This command can select only the normal view, which appears on the color monitor or in the left display box on the monochrome monitor.

Create a New View--Creates a new view file. The actual view must be created with the graphics editor. When a new view is created, the user must specify the following information:

- View name
- View physical path

Find-View--Finds and loads a view from the disk.

Save All Modified Views--Saves all views that have been updated.

Save All Views--Saves all views.

Display View States--Displays the current state of all known views, indicating whether the view is loaded and whether it has been modified since being loaded.



**view item commands**

Change View Attributes--Changes the attributes for the current view. The following attributes can be changed:

- View name
- View physical path

Save View--Saves the current view.

Kill View--Removes a view. This is the opposite of loading a view.

Probe View--Updates the icons in the currently displayed views with the values from the associated state variables in the model.

**GRAPHICS EDITOR**

The Graphics Editor is used to create and edit views. See Figure 1 (text).

**View Commands**

Create--Creates a new view.

Select--Selects an existing view for editing.

Save--Acts according to which mouse button is pushed:

- Save the Current View
- Save All System Views

Attribute--Alters the attributes of the current view.

Kill--Kills the current view. This is the opposite of loading a view.

Probe--Updates the icons in the current view with the values from the associated state variables in the model.

Flavor--Makes the current view into a flavor.

List--Lists all icons in the current view.

Reorder--Reorders the icons in the current view for drawing and probing.

Taps--Makes a file of all taps in the current view.

Draw--Clears the screen and redraws the current view.



## GRAPHICS EDITOR

## View Commands

Hardcopy--Acts according to which mouse button is pushed:

- Hardcopy Color Screen--Produces a hardcopy printout of the view displayed on the color monitor.
- Set up Hardcopy Options--Changes hardcopy output options.
- Preview Hardcopy Image.

Interact--Interacts with the current view. The cursor is moved to the current view on the color monitor (or the left portion of the monochrome monitor), and the user clicks the mouse while the cursor is positioned over a graphic icon. If the icon has an output tap (set tap) into the model, the state variable associated with the icon is changed.

Initialize--Acts according to which mouse button is pushed:

- Initialize Icon Graphics System
- Initialize System Model Process

Configure--Displays a menu of configurations:

- Normal--Causes normal views to be displayed on the color monitor.
- B&W--Causes normal views to be displayed in the left display box on the monochrome monitor.

## Simenv Commands

Activity--Allows selection of different processes:

- Model Control (Simenv)
- Lesson Editor

## Mark Commands

These commands are used to mark one or more icons in the view that is being edited.

Highlight--Highlights the marked icons.

Clear--Clears highlighting of the marked icons.

Mark--Marks one or more icons with the mouse. This command moves the cursor to the current view on the color monitor to allow the user to mark or unmark specific icons with the mouse.

All--Marks all icons in the current view.



## GRAPHICS EDITOR

## Mark Commands

Tapped--Acts according to which mouse button is pushed:

- Mark Tapped Icons--Marks all icons in the current view that have input taps (probe taps) defined. These icons are updated when the model state variable change.
- Mark Tap-set Icons--Marks all icons in the current view that have output taps (set taps) defined. These icons can be modified through interaction with the view. The interaction causes the associated state variables in the model to be modified.

Untapped--Marks all icons in the current view that do not have either input or output taps defined.

Type--Marks all icons of a specified type.

Find--Finds and marks a named icon in the current view.

Misc--Miscellaneous mark commands.

### Edit Marked Icon Commands

These commands are used to edit icons that have been marked.

Delete--Deletes all marked icons.

Undelete--Undoes the previous deletion.

Probe--Updates the marked icons in the current view with the values from the associated state variables in the model.

Default--Acts according to which mouse button is pushed:

- Store Default Parameters--Stores the marked icon's parameters as the default parameters for icons of that type.
- Remove Default Parameters--Removes default parameters for specified icon types.

List--Lists the names for all marked icons.

Draw--Draws the marked icons.

Describe--Describes the marked icons; produces a listing of all of the parameters associated with each marked icon.

Inspect--Allows the operator to inspect the marked icons; activates the flavor examiner to allow inspection of the flavors making up the marked icons.

Move--Repositions the marked icons as a group without changing their size.



## GRAPHICS EDITOR

## Edit Marked Icon Commands

Copy--Copies the marked icons.

Edit--Edits the shape of the marked icons by adjusting their sides or corners.

Shape--Changes the position and size of the marked icons. Pressing the mouse changes the shapes of all marked icons, one at a time.

Rotate--Allows marked icons to be rotated. Each marked icon is rotated individually.

Reflect--Reflects the region containing the marked icons about a horizontal or vertical axis.

Tap--Allows the operator to specify input taps (probe taps) and output taps (set taps) for an icon.

Name--Allows the operator to specify a name for an icon.

Color--Changes the color of marked icons.

Label--Provides a label for an icon.

Picture--Selects a picture for marked icons. (This command appears to be broken.)

Miscellaneous--Adjusts miscellaneous parameters for an icon.

Inter View--Allows the operator to copy the marked icons into a buffer or the buffer into the current view.

## Grid Commands

These commands are used to draw a grid for positioning icons in a view.

Draw--Draws a grid on the screen.

Show--Toggles the grid between visible and invisible. (This command appears to be broken.)

Size--Changes the spacing between dots on the grid.

Points--Forces the cursor to move only to grid points when manipulating icons on the color monitor.

Diagonal--Forces the cursor to move only at 45 degrees from the horizontal when manipulating icons on the color monitor.

T Square--Forces the cursor to move only in horizontal or vertical directions when manipulating icons on the color monitor.



## GRAPHICS EDITOR

### Icon Selection

New icons are added to a view by selecting one of the following icon types and positioning it on the color monitor with the cursor.

Circle	Rectangle
Lozenge	Triangle
Trapezoid	Diamond
Hexagon	Octagon
Line	Spline
Polygon	Text
Banner	Graph
Multi Plot Graph	Dial
Column	Tank
Digital Bar	Force Bar
Bar	Signal
Flame	Pipe
Centrifugal Pump	Rotary Pump
Air Ejector	Y Strainer
Duplex Strainer	Impulse Trap
Orifice	Sstg
Ssdg	Circuit Breaker
Fusible Link	Fuse
Biscuit	Bar Switch
Knife Switch	Rotary Switch
Toggle Switch	Stop Valve
Anglestop Valve	Check Valve
Relief Valve	Safety Valve
Regulator Valve	3 Way Valve
4 Way Valve	

### Status Line Commands

All of the status line commands available under model control are also available under the graphics editor.



## **LESSON EDITOR**

The lesson editor (Figure B-2) is used to create and edit lessons. A lesson is a sequence of instructions and text that is used to explain the operation of the simulation.

### **Lesson Commands**

Select--Selects an existing lesson.

Find--Finds and loads a lesson from disk.

Create--Creates a new lesson.

Save--Saves the current lesson.

Kill--Kills the current lesson. This is the opposite of loading a lesson.

Attribute--Edits the attributes for the current lesson.

Play--Plays back the current lesson. This executes the sequence of instructions (segments) that are defined for the current lesson.

Help ?--An unimplemented feature that could be used to display help information.

### **View Commands**

Select--Acts according to which mouse button is pushed:

- Select a View--Selects a defined view and displays it either on the color monitor or in the display box on the monochrome monitor.
- Selects a Control View--Selects a defined view and displays it in the display box on the monochrome monitor. For the display to appear, the monochrome monitor must be configured for Control Normal display.

Initialize--Acts according to which mouse button is pushed:

- Initialize Icon Graphics System.
- Initialize System Model Process.

Configure--Displays a menu of configurations:

- Normal--Causes normal views to be displayed on the color monitor. Control views are not displayed.
- Control Normal--Causes normal views to be displayed on the color monitor. Control views can be displayed in the display box on the monochrome monitor.
- B&W--Causes normal views to be displayed in the display box on the monochrome monitor. Control views are not displayed.





Lesson Editor				
Lesson Select Find Create	Save Kill Attribute	Play Help?	Select Initialize Configure	Activity <i>Simenv</i>
View Highlight Icons Run Icon Icon Condition Reset <i>Segments</i>			Text Stop Pause <i>Edit</i>	Perform Delete Undelete Copy Edit Name
Lesson : None <i>Beginning of Lesson</i>				
<i>display window</i>			<i>End of Lesson</i>	
System: Steamer System    Model: Steamer Model [stopped]			Sub System: All    View: Main Engine Lube Oil	
				
[Fri 27 May 1:05:30] 11spn		SIM: <u>User Input</u>		

Figure B-2. Lesson editor interface.



## LESSON EDITOR

### Simenv Commands

Activity--Allows selection of different processes:

- Model Control
- Graphics Editor

### Segment Commands

The segment commands are used to add new segments to the lesson. Each segment is a specific instruction that performs an operation. The segment commands are also used to perform view operations. The window that contains the segment commands is small, and all commands cannot appear in the window simultaneously. To see all of the commands, the cursor must be moved to the left edge of the window to scroll the commands.

Reset--Acts according to which mouse button is pushed:

- Add a Reset Model Segment to the Lesson--This type of segment causes the model to be reset.
- Reset the Model.

Run--Acts according to which mouse button is pushed:

1. Add a Run Segment to the Lesson--This type of segment runs the model when executed.
2. Run the Model.

Stop--Acts according to which mouse button is pushed:

- Add a Stop Segment to the Lesson--This type of segment stops the model when executed.
- Stop the Model.

Rate--Acts according to which mouse button is pushed:

- Add a Set Model Rate Segment to the Lesson--This type of segment causes the tick rate for the model to change.
- Set the Model Rate.

Tick--Acts according to which mouse button is pushed:

- Add a Tick Segment to the Lesson--This type of segment runs the model for one tick.



## 3. Run the Model for 1 Tick.

Status--Acts according to which mouse button is pushed:

- Add a Status Segment to the Lesson--This type of segment displays the current model status on the monochrome monitor.
- Show the Model Status.

View--Acts according to which mouse button is pushed:

- Add a Normal View Segment to the Lesson--This type of segment selects a new view to be displayed as the normal view. (This command causes a system error when executed.)
- Add a Control View Segment to the Lesson--This type of segment selects a new view to be displayed as the control view. (This command causes a system error when executed.)

## 4. Select a New View--Selects a new normal view.

Text--Adds a text segment to the lesson. This type of segment causes text to be displayed on the monochrome monitor when executed.

Pause--Adds a pause segment to the lesson. This type of segment causes the lesson to pause for a specified number of seconds (default is 5 seconds).

Mouse Pause--Adds a mouse pause segment to the lesson. This type of segment causes the lesson to pause until the operator clicks a mouse button within a displayed window.

Icon--Adds an icon behavior segment to the lesson. This type of segment changes the state of an icon that has an output tap defined in the normal view or the control view.

Icon Condition--Adds an icon conditional segment to the lesson. (Execution of this command causes a system trap to occur. Its intended operation is unknown.)

Highlight Region--Adds a highlight region segment to the lesson. This type of segment causes a region or regions on the screen to be highlighted. The remainder of the screen is masked in gray. Highlighting can be shown only on the color monitor.

Highlight Icons--Adds a highlight icon segment to the lesson. This type of segment causes specific icons to be highlighted. The remainder of the screen is masked in gray. (This command did not work, as no action occurred when it was selected.)

Clear Highlight--Adds a clear highlight segment to the lesson. This type of segment clears any highlighting on the screen.

Function--Adds a function segment to the lesson. This type of segment executes a LISP function. (It is not know if this command works.)



## **LESSON EDITOR**

### **Edit Commands**

The edit commands are used to edit segments that have been created using the segment commands. This is done by highlighting the segment to be edited and then selecting the appropriate edit command.

**Perform--**Executes the highlighted segment(s) once.

**Delete--**Deletes the highlighted segment(s).

**Undelete--**Undeletes previously deleted segments. Segments can be moved by deleting them and then undeleting them in a different location in the lesson sequence.

**Copy--**Copies the highlighted segment(s). (Nothing happens when this command is selected.)

**Edit--**Edits parameters for the highlighted segment(s). For example, the number of seconds for a pause segment can be changed. Not all segments have parameters that can be edited.

**Name--**Changes the description for a lesson segment.

### **Status Line Commands**

All of the status line commands available under model control are also available under the lesson editor.

### **Lesson Status Commands**

A second status line is shown that gives lesson status. Commands can be chosen by positioning the cursor over one of the selections on the status line and clicking one of the mouse buttons. The status line selections consist of "Lesson" and the currently selected lesson.

### **lesson commands**

**Select a Lesson--**Selects an existing lesson.

**Create a New Lesson--**Creates a new lesson.

**Find Lesson--**Finds and loads a lesson from the disk.

**Save All Lessons--**Saves all current lessons onto the disk.

**Display Lesson States--**Displays the current state of all known lessons, indicating whether the lesson is loaded and whether it has been modified since being loaded.



## LESSON EDITOR

## Lesson Status Commands

### lesson item commands

Change Lesson Attributes--Changes the attributes for the current lesson.

Save Lesson--Saves the current lesson.

Kill Lesson--Removes the current lesson. This is the opposite of loading a lesson.

### OPERATING SYSTEM COMMANDS

Each of the Steamer prototype screens provides the operator with a window in which to enter the operating system commands for the Symbolics machine. Since these commands are fully described in the Symbolics documentation, they are not discussed here.



**APPENDIX C**  
**STEAMER PROTOTYPE GLOSSARY**



## **Attributes**

Attributes are operating parameters that are defined for each system, subsystem, model, view, and lesson. These attributes consist of (1) a name for the system, subsystem, model, view, or lesson and (2) a location on the disk where the data base for that item is stored. Depending on the item, attributes may also include other definable parameters.

## **Black-and-White (or monochrome) Monitor**

The black-and-white monitor is one of two monitors used for Steamer. Commands that are selected with the mouse and views are presented to the user on the monitor.

## **Casualty**

Casualties are model malfunctions that are initiated by the user. For example, the user can initiate a casualty that would force a pump to fail. The rest of the model reacts to the casualty and the user observes the result. By watching subsequent system indicators such as tank levels and dial readings, the user gains an understanding of the problems created by the casualty.

## **Color Monitor**

The color monitor is one of two monitors used for Steamer. It is used to display views.

## **Control View**

Steamer can display two views simultaneously. The control view is the secondary view and is displayed on the monochrome monitor. The primary view is displayed on the color monitor. Any Steamer view can be placed in the control view.

## **Display Box**

The display box is a window on the monochrome monitor that displays views.

## **Flavor**

A flavor is part of Symbolics' object-oriented extension to the ZetaLisp programming language. Flavors represent objects within the language.

## **Flavor Examiner**

The flavor examiner is a Symbolics utility that is used to examine flavors that have been created.



## **Graphics Editor**

The graphics editor is the portion of Simenv that is used to create and modify views. Using the graphics editor, the user builds a view by selecting and positioning icons within the view. The icons can be tapped into the model using input taps and output taps.

## **Grid**

The grid is a tool for positioning icons when using the graphics editor. It consists of a series of dots that are spaced at regular intervals on the color monitor. When creating views, the user can use the grid points to line up icons or use positioning commands that will force lines to terminate on grid points.

## **Highlighting**

Highlighting is a method for enhancing part of a graphic view so that it stands out from the rest of the view. For Simenv, this is done by placing a gray mask over the portion of the view not being highlighted, making the unmasked portion of the view stand out. Highlighting can be done only on the color monitor.

## **Icon**

An icon is a graphic symbol that can be placed in a view. Simple icons are used to create shapes such as rectangles, circles, and lines. More complex icons depict steam plant components such as valves, pipes, pumps, and flames. Icons are also available for creating graphs and text. Icons placed in a view can be connected to state variables in the model using taps. As the value of a tapped state variable changes, the icon changes to show the new value graphically.

## **Input Tap**

This tap connects an icon and a state variable. If the state variable changes, the icon is updated to show the change graphically. An input tap is also called a probe tap.

## **Interaction**

Interaction is the process of changing the values of icons within a view. A user interacts with a view by selecting the interact command. The cursor is moved to the normal view, and the operator positions the cursor over an icon in the view and clicks the left mouse button. If an output tap is defined for the icon, the value of the icon is changed and the value of model state variable that is defined for the output tap is updated. The user clicks the middle mouse button to terminate interaction.

## **Kill**

The kill command removes an item so that Simenv is no longer aware of it. Killing an item does not delete it from the disk, it simply removes it from the list of items that Simenv can address. The item can later be reloaded from disk using the load command.



## **Lesson**

A lesson is a series of segments (instructions) that manipulate the simulation to illustrate a concept or a series of actions. Each segment within the lesson causes a change to the model or to a view. Text can be added to explain the change that is made and to describe the results that the change will cause in the simulation. When a lesson is run, Simenv sequences through the segments in order, creating a "story" that describes the operation of the device being simulated.

## **Lesson Editor**

The lesson editor is used to create and modify lessons. See Lesson.

## **Load**

The load command retrieves part of the system data base from disk so that Simenv can address it. Loading an item does not automatically select it.

## **Malops**

Malops is an abbreviation for malfunctioning operation. Whenever the value of any state variable in the model is outside its normal operating range, a malops is signaled.

## **Model**

A model is an algorithm that mathematically simulates the operations of a physical device. The model maintains a set of state variables. The state variables correspond to components within the actual physical device, and their values reflect possible values of the device components (for example: on/off, 200 psi, etc.). When the model is operating, it updates the state variables to reflect actual operation of the physical device.

Each model must have defined routines that Simenv can access to reset the model to an initial condition, run the model, and get the status of the model.

## **Model Control**

Model control is the portion of Simenv that creates and controls a simulation. Model control is used to create and modify systems, models, and subsystems. It is also used to run the model, reset the model, and change model parameters. Model control commands are used to select views for display and to interact with the views.

## **Output Tap**

An output tap connects an icon and a state variable so that, when the icon is changed (through user interaction), the value of state variable is updated. An output tap is also called a set tap.

## **Primary View**

The primary view is one of the two views that Simenv can display simultaneously. The primary view is displayed on the color monitor. Any Steamer view can be placed in the primary view.



**Probe**

The probe command is used to update the icons in the current view that have input taps defined. A probe causes the icons to reflect the values of the associated model state variables.

**Probe Tap**

See Input Tap.

**Rate**

The rate determines the relationship between time real and model operating time. For a rate of 1, the model operates at real time. For a rate of 0.1, it operates 10 times faster than real time.

**Segment**

A segment is a single instruction within a lesson. It manipulates the simulation. Segments are provided to start and stop the model, display different views, modify the state of view icons, and present text that describes the changes that are taking place.

**Set Tap**

See Output Tap.

**Simenv**

Simenv is short for simulation environment. Simenv is the program that creates and manipulates simulations. Using Simenv, a user creates the models and views that simulate a physical device. Once the simulation is created, Simenv is used to operate the simulation and look at views of the simulation. Simenv is composed of a model control facility and a graphics editor.

**State Variable**

A state variable is a variable within the model that represents a component in the physical device. The values of the state variable correspond to values of the actual device component. For example, a state variable that represents a switch has the possible values of on and off. A state variable that represents a steam pressure might have a value of 200 psi.

**Steamer System**

Steamer system is the collection of code (the model) and graphic pictures (views) that simulates operation of the 1200-psi steam plant.

**Subsystem**

A subsystem is a collection of views for a system. Once a number of views have been created, they can be grouped into various subsystems based on the user's criteria.



## **System**

A system is a simulation that was built using the simulation environment (Simenv). For example, the simulation of the 1200-psi steam plant is a system called Steamer system. Each system is composed of one or more models, one or more subsystems, and one or more views.

## **Tap**

A tap is the connection between a view icon and a model state variable. Two types of taps are provided. An input tap (or probe tap) that connects an icon and a state variable so that, if the value of the state variable changes, the icon is updated graphically to show the change. An output tap (or set tap) connects an icon and a state variable so that, if the icon changes (through user interaction), the value of the state variable is updated. Each icon can have both an input tap and an output tap.

## **Tick**

A tick is one execution of the simulation model. If the model is run for one tick, all of the model routines are executed once and the state variables are updated once.

## **Untapped icons**

Untapped icons are icons that have no defined input or output taps.

## **View**

A view is a graphical representation of a portion of a simulated device. Each view is composed of graphical icons that correspond to actual components in the simulated device. The icons are arranged in the view to show connection between the device components. A view can be displayed on either the color or monochrome monitor.



## DISTRIBUTION LIST

Chief of Naval Operations (OP-11B1), (OP-01B2)  
Chief of Naval Technical Training  
Surface Warfare Officers School Pacific, Coronado  
Surface Warfare Officers School Command, Newport  
Officer in Charge, Engineering Systems Schools, Service Schools Command, Naval Training Center, Great Lakes  
Office of Naval Technology (Code 222)  
Office of Naval Research (OCNR-1142CS)  
Commander Naval Reserve Force  
Commanding Officer, Naval Education and Training Program Management Support Activity (Code 03) (2)  
Naval Training Systems Center  
Army Research Institute (PERI-POT-I)  
Air Force Human Resources Laboratory (IDI)  
Defense Technical Information Center (DTIC)